

Отчёт о прохождении внешнего курса

часть 3

Дисциплина: Операционные системы

Мишина Анастасия Алексеевна

Содержание

1 Цель работы	6
2 Задание	7
3 Выполнение лабораторной работы	8
4 Глава 3.1 Текстовый редактор vim	9
5 Глава 3.2 Скрипты на bash: основы	14
6 Глава 3.3 Скрипты на bash: ветвления и циклы	17
7 Глава 3.4 Скрипты на bash: разное	26
8 Глава 3.5 Продвинутый поиск и редактирование	33
9 Глава 3.6 Строим графики в gnuplot	39
10 Глава 3.7 Разное	43
11 Вывод	47

Список иллюстраций

4.1	Назначения удаленного сервера	9
4.2	Выполнение задания	9
4.3	Выбор подходящих ответов	10
4.4	Выполнение задания	10
4.5	Выбор подходящих ответов	11
4.6	Объяснение	11
4.7	Ввод ответа	12
4.8	Режим выделения	12
4.9	Нерабочее соединение	13
5.1	Наборы команд	14
5.2	Пути к файлу	15
5.3	Варианты названия переменных	15
5.4	Написание программы	16
6.1	Правила	17
6.2	Правила	18
6.3	Варианты условий	19
6.4	Код программы в редакторе	20
6.5	Тестирование кода с аргументами 3 и 5	20
6.6	Ввод ответа на степике	21
6.7	Задание	22
6.8	Программный код	22
6.9	Программный код	23
6.10	Выполнение программы	23
6.11	Ввод ответа на степике	24
6.12	Задание	25
6.13	Ввод ответа на степике	25
7.1	Сложение переменных	26
7.2	Варианты сложения	27
7.3	Вывод пути	27
7.4	Задание и ответ	28
7.5	Скрипт с функцией counter	28
7.6	Выполнение скрипта и ответ	29

7.7	Задание и ответ	29
7.8	Задание	30
7.9	Алгоритм из моего скрипта	30
7.10	Сам скрипт	31
7.11	Задание	32
7.12	Мой скрипт	32
8.1	Список файлов	33
8.2	Работа опций команды find	34
8.3	Поиск	35
8.4	Размер файлов results.txt	36
8.5	Объяснение	36
8.6	Задание и ответ на степике	37
8.7	Вывод команды sed	37
8.8	Задание на степике	38
8.9	Ответ на степике	38
9.1	Опция gnuplot	39
9.2	Обозначения графиков	40
9.3	Оформление точек	40
9.4	Начальная программа	41
9.5	Итоговая программа	42
10.1	Права доступа	43
10.2	Объяснение решения	44
10.3	Задание	44
10.4	Права доступа	45
10.5	Что может команда wc?	45
10.6	Опции du	46
10.7	Быстрое создание нескольких директорий	46
11.1	Прохождение курса “Введение в Lunix”	47

Список таблиц

1 Цель работы

Приобретение практических навыков работы с операционной системой
Линукс. Продвинутые темы.

2 Задание

- 3.1 Текстовый редактор vim
- 3.2 Скрипты на bash: основы
- 3.3 Скрипты на bash: ветвления и циклы
- 3.4 Скрипты на bash: разное
- 3.5 Продвинутый поиск и редактирование
- 3.6 Строим графики в gnuplot
- 3.7 Разное

3 Выполнение лабораторной работы

4 Глава 3.1 Текстовый редактор vim

Первое задание в этой главе (рис. [4.1]). Ответ на него в видеоуроке (двоеточие для ввода команды, q - quit).

3.1 Текстовый редактор vim 5 из 13 шагов пройдено 1 из 10 баллов получен

Какую клавишу(и) нужно нажать на клавиатуре, чтобы выйти из редактора vim? Считайте, что вы только что открыли файл и вам сразу понадобилось выйти из редактора.

Выберите один вариант из списка

Правильно, молодец!

Верно решили 32 523 учащихся
Из всех попыток 69% верных

"Ctrl", затем "X"
"Esc"
 ":" , затем "q", затем "Enter"
".", затем "q"
"q", затем "Enter"

Следующий шаг Решить снова

Ваши решения Вы получили: 1 балл из 1

1029 402 Шаг 5 Следующий шаг >

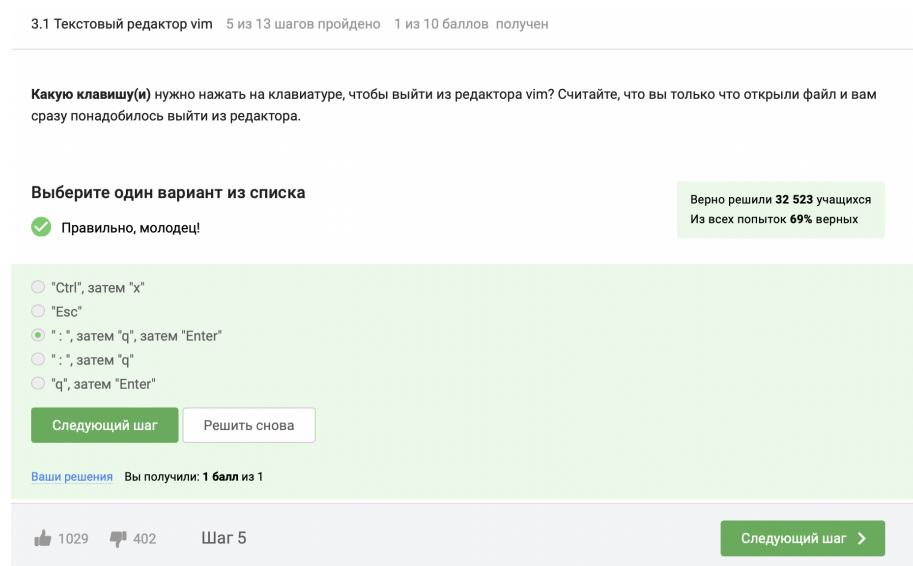
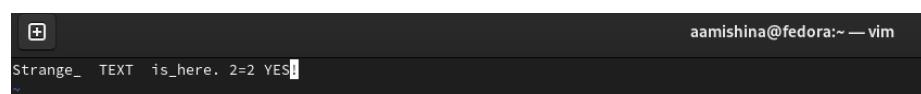


Рис. 4.1: Назначения удаленного сервера

Следующее задание учит нас работе с кодовыми клавишами в редакторе vim. Для его выполнения я создала файл, записала в него строку и использовала сочетания в примерах, чтобы ответить на вопросы (рис. [4.2]), (рис. [4.3]).



```
aamishina@fedora:~ — vim
Strange_TEXT_is_here. 2=2 YES!
```

Рис. 4.2: Выполнение задания

При перемещении в vim "по словам" есть небольшая разница в том, используем мы маленькую (`w`, `e`, `b`) или большую (`W`, `E`, `B`) букву. Первые перемещают нас по "словам" (`word`), а вторые по "большим словам" (`WORD`). Посмотрите справку по этим перемещениям и разберитесь в чем заключается разница между `word` и `WORD`.

А для того, чтобы убедиться, что вы разобрались, отметьте ниже **все верные** утверждения про следующую строку:

`Strange_ TEXT is_here. 2=2 YES!`

Примечание: во всех утверждениях имеется ввиду, что мы находимся в редакторе vim, включен нормальный режим работы и курсор находится в самом начале строки.

Подсказка: чтобы вызвать vim-справку по, например, перемещению `w`, нужно открыть vim и ввести команду `:help w`. Вы попадете в то место справки, где описано это перемещение, а так как все перемещения описаны рядом, то двигаясь по тексту вверх и вниз можно прочитать и про `e` и про `b` и, самое главное, про `word` и `WORD`. Кроме того, можно вызывать сразу справку по термину `word` при помощи `:help word`. Чтобы закрыть справку, нужно ввести команду `:q`.

Выберите все подходящие ответы из списка

Абсолютно точно.

Верно решили 25 385 учащихся
Из всех попыток 20% верных

- Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).
- В этой строке 5 "слов" (`word`)
 - Чтобы попасть в конец строки, нужно одинаковое число нажатий, что на `W`, что на `w`
 - Нажимая только на `w`, нельзя переместить курсор на ""
 - После 10 нажатий на `W` курсор окажется там же, где бы он был после 10 нажатий на `w`
 - В этой строке 9 "больших слов" (`WORD`)
 - В этой строке 9 "слов" (`word`)

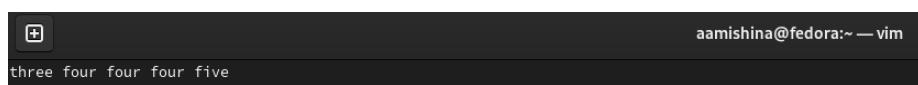
[Следующий шаг](#)

[Решить снова](#)

Рис. 4.3: Выбор подходящих ответов

В видеоуроке рассказывалось об основных клавишиах. Например, `de` - удалить до конца текущего слова, `d$` - удалить до конца строки, `d5w` - удалить 5 слов, `dd` - удалить строку, `d10d` - удалить 10 строк, `i` - войти в `insertion mode`, `a` - сдвинуть курсор вправо и войти в `insertion mode`, `o` - вставить пустую строку снизу и редактировать ее, `O` - то же, но строка сверху, `y` - скопировать, `yy` - скопировать строку, `p` - вставить перед, `P` - вставить после.

Это задание также выполняем в редакторе (рис. [4.4]), (рис. [4.5]).



```
aaamishina@fedora:~ — vim
three four four four five
```

Рис. 4.4: Выполнение задания

Предположим, что в текстовом файле записана одна единственная строка:

one two three four five
и вам нужно преобразовать её в строку
three four four four five

Какие(ой) из предложенных ниже наборов нажатий клавиш выполняют такое редактирование? В этих наборах нажатие на клавишу Esc обозначается как <Esc> (т.е. знаки "<" и ">" не несут отдельного смысла).

Примечание: во всех утверждениях имеется в виду, что мы находимся в редакторе vim, включен нормальный режим работы и курсор находится в самом начале строки.

Выберите все подходящие ответы из списка

Верно. Так держать!

Верно решили 23 655 учащихся

Из всех попыток 16% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

- ddithree four four four five<Esc>
- d2wwfour four <Esc>
- d2w\$bfour four <Esc>
- d2wwywPp
- x2wwywPp
- d2wwywpp

[Следующий шаг](#)

[Решить снова](#)

Ваши решения Вы получили: 1 балл из 1

Рис. 4.5: Выбор подходящих ответов

Объяснение следующего задания представлено на картинке (меняем Windows на Linux) (рис. [4.6]), (рис. [4.7]).

:%s/<что меняем>/<на что меняем>|

Рис. 4.6: Объяснение

Предположим, что вы открыли файл в редакторе vim и хотите заменить в этом файле все строки, содержащие слово `Windows`, на такие же строки, но со словом `Linux`. Если в какой-то строке слово `Windows` встречается больше, чем один раз, то заменить на `Linux` в этой строке нужно **только самое первое** из этих слов.

Какую команду нужно ввести для этого в vim? Укажите необходимую команду целиком (т.е. **включая** ввод ":" в самом начале), однако нажатие на `Enter` после ввода команды обозначать никак **не нужно**.

Напишите текст

Правильно.

Верно решил **24 631** учащийся
Из всех попыток **57%** верных

:%s/Windows/Linux

Следующий шаг

Решить снова

Ваши решения Вы получили: **2 балла из 2**

Рис. 4.7: Ввод ответа

Для решения этого задания обратимся к информации в интернете про режим выделения Visual в vim. Стоит отметить, что режим выделения открывается только из нормального режима, а также для выхода надо использовать двойное нажатие клавиши esc (рис. [4.8]).

Мы совсем не рассказали вам про третий режим работы vim – режим **выделения (Visual)**. Предлагаем вам ознакомиться с ним самостоятельно. Например, это можно сделать во время прохождения упражнений в vimtutor, который мы настоятельно рекомендуем вам для изучения vim!

Чтобы убедиться, что вы разобрались с этим режимом работы, отметьте, пожалуйста, **все верные** утверждения из списка ниже.

Подсказка: если вы не хотите проходить vimtutor целиком, то можете открыть его и поиском найти слово "**Visual**". Вы попадете в задание, прохождение которого будет вполне достаточно, чтобы выполнить это задание.

Выберите все подходящие ответы из списка

Всё получилось!

Верно решили **23 497** учащихся

Из всех попыток **26%** верных

Количество успешных решений
(обновляется каждые 2 мин)

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

Чтобы выйти из режима выделения, нужно ввести :q

В режиме выделения можно использовать команды перемещения (например, W, e, \$, и др.)

В режиме выделения можно использовать команды d (удалить) и u (скопировать)

Выйти из режима выделения можно, нажав клавишу Esc два раза

Режим выделения открывается из нормального режима по нажатию "v"

Режим выделения открывается из любого другого режима по нажатию "v"

Следующий шаг

Решить снова

Ваши решения Вы получили: **2 балла из 2**

Рис. 4.8: Режим выделения

Финальное задание в этой главе было на работу с удаленным сервером.

ром. Терминал степика снова не смог установить соединение с сервером, прикрепляю фотографию с доказательством (рис. [4.9]).

```
box@69fcdf2e01bf: ~ $ wget https://stepik.org/media/attachments/course73/byron.t  
xt.gz  
--2023-05-20 14:34:09-- https://stepik.org/media/attachments/course73/byron.txt  
.gz  
Resolving stepik.org (stepik.org)... failed: Temporary failure in name resolution.  
wget: unable to resolve host address 'stepik.org'  
box@69fcdf2e01bf: ~ $
```

Рис. 4.9: Нерабочее соединение

5 Глава 3.2 Скрипты на bash: основы

Так как мы остаемся в новой третьей оболочке, которая не будет видеть команды предыдущих оболочек, перелистываться будет только набор С (рис. [5.1]).

The screenshot shows a programming challenge interface. At the top, it says "3.2 Скрипты на bash: основы" and "3 из 10 шагов пройдено 1 из 6 баллов получен". The main text asks: "Надеемся, что вы разобрались, что одну оболочку (например, sh) можно запустить из другой оболочки (например, из bash). Предположим, что вы открыли терминал и у вас в нем запущена оболочка bash. Вы набираете в ней команды A1, A2, A3, а затем запускаете оболочку sh. В этой оболочке вы набираете команды B1, B2, B3 и запускаете оболочку bash. И, наконец, в этой последней оболочке вы набираете команды C1, C2, C3. Если теперь вы попробуете при помощи стрелочек вверх/вниз перемещаться по истории набранных команд, то команды из какого набора(ов) будут появляться?"

Below the text is a question: "Выберите один вариант из списка". There are five options with radio buttons:

- Из наборов А и С
- Только из набора С
- Только из набора А
- Только из набора В
- Никакие команды появляться не будут

A green checkmark icon next to the second option indicates it is correct. Below the list are two buttons: "Следующий шаг" and "Решить снова". To the right, a box shows statistics: "Верно решили 30 266 учащихся" and "Из всех попыток 65% верных". At the bottom, it says "Ваши решения Вы получили: 1 балл из 1". The footer shows "1036" likes and "130" dislikes, and the step number "Шаг 3".

Рис. 5.1: Наборы команд

Как видно из кода программы, для начала мы перейдем в директорию /home/bi, где и создадим файл file1.txt. Соответственно, путь до файла будет /home/bi/file1.txt (рис. [5.2]).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [script1.sh](#), [script2.sh](#).

Предположим, что вы находитесь в директории `/home/bi/Documents/` и запускаете в ней скрипт следующего содержания:

```
#!/bin/bash
cd /home/bi/
touch file1.txt
cd /home/bi/Desktop/
```

Как будет выглядеть **абсолютный путь** до созданного файла `file1.txt` по окончанию работы скрипта?

Выберите один вариант из списка

Верно.

Верно решили **29 905** учащихся
Из всех попыток **76%** верных

- /home/bi/file1.txt
- /home/bi/Documents/file1.txt
- Никак (файла file1.txt не будет существовать после завершения работы скрипта)
- /home/bi/Desktop/file1.txt

[Следующий шаг](#)

[Решить снова](#)

Ваши решения Вы получили: **1 балл из 1**

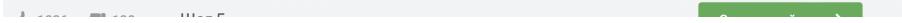


Рис. 5.2: Пути к файлу

Из видеоурока мы узнаем, что нельзя использовать точки, тире и слэш в названиях переменных, также они не могут начинаться с цифр (рис. [5.3]).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [variables1.sh](#), [variables2.sh](#).

Какие из представленных ниже строк **могут** быть именами переменных в bash? Выберите **все** подходящие варианты!

Подсказка: если все варианты ответов являются неверными, то не отмечайте ни один из них и нажмите кнопку "Отправить"/"Submit".

Выберите все подходящие ответы из списка

Отлично!

Верно решили **27 188** учащихся
Из всех попыток **25%** верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

- variable123
- variable
- var-i-able
- _variable
- var.i.able
- _variable
- VARiable

[Следующий шаг](#)

[Решить снова](#)

Ваши решения Вы получили: **1 балл из 1**

Рис. 5.3: Варианты названия переменных

Финальное задание мини-главы требовало написания легкого скрипта на баш, который бы принимал на вход два значения и выводил их. Для

начала записываем введенные аргументы в переменные, затем в одной строчке выводим их в сообщении (рис. [5.4]).

Вы можете скачать и изучить скрипт, который мы показали в видеофрагменте: [arguments.sh](#).

Напишите скрипт на bash, который принимает на вход два аргумента и выводит на экран строку следующего вида:

```
Arguments are: $1=первый_аргумент $2=второй_аргумент
```

Например, если ваш скрипт называется `./script.sh`, то при запуске его `./script.sh one two` на экране должно появиться:

```
Arguments are: $1=one $2=two
```

а при запуске `./script.sh three four` будет:

```
Arguments are: $1=three $2=four
```

Подсказка: в случае проблем с решением задачи, обратите внимание [на наши рекомендации по написанию скриптов](#).

Напишите программу. Тестируется через stdin → stdout

✓ Всё получилось!

Верно решили 25 053 учащихся

Из всех попыток 41% верных

Теперь вам доступен [Форум решений](#), где вы можете сравнить свое решение с другими или спросить совета.

```
1 #!/bin/bash
2
3 first=$1
4 second=$2
5 echo "Arguments are: \$1=$first \$2=$second"
6
7
8
9
```

Следующий шаг

Решить снова

Рис. 5.4: Написание программы

6 Глава 3.3 Скрипты на bash: ветвления и циклы

Выбираем в задании все варианты, которые правдивы при любых значениях переменных. В моем случае всегда верны все ответы. Для решения пользуемся следующими правилами (рис. [6.1]), (рис. [6.2]).

Ветвление: основы

Условия (файлы):

```
-e <путь> # путь существует  
-f <путь> # это файл  
-d <путь> # это директория  
-s <путь> # размер файла больше 0  
-x <путь> # файл исполняемый
```

Условия (логические):

```
! # отрицание логического выражения  
&& # логическое «И»  
|| # логическое «ИЛИ»
```

Рис. 6.1: Правила

Ветвление: основы

Условия [числа (строки)]:

<число/строка> операция <число/строка>

```
-eq, (==) # равно
-ne, (!=) # не равно
-lt, (<) # меньше
-le,      # меньше или равно
-gt, (>) # больше
-ge,      # больше или равно
```

Рис. 6.2: Правила

Объяснение решения (рис. [6.3]):

- -z “” - всегда верно, строка пуста
- !(4 -le 3) - всегда верно, 4 не меньше либо равно 3
- -s \$0 - всегда верно, размер файла больше 0
- -n \$0 - всегда верно, строка не пуста
- либо переменные равны, либо не равны
- 5 -ge 5 - всегда верно, 5 больше или равно 5

Вы можете скачать и изучить скрипт, который мы показали в видеофрагменте: [branching1.sh](#).

Предположим, вы пишете скрипт на bash и хотите использовать в нем конструкцию `if` в следующем фрагменте:

```
if [[ ... ]]  
then  
    echo "True"  
fi
```

Вы можете вписать вместо "...": (внутри `[[]]` и не забудьте про пробелы после `[` и перед `]]`!) любое из перечисленных ниже условий. Однако мы просим вас выбрать только те из них, при которых `echo` напечатает на экран `True` вне зависимости от того, с какими параметрами был запущен ваш скрипт и какие в нем есть переменные.

Например, условие `0 -eq 0` **подходит**, т.к. ноль всегда равен нулю вне зависимости от аргументов и переменных внутри скрипта и на экран будет напечатано `True`. В то же время условие `$var1 -eq 0` **не подходит**, так как в переменной `var1` как может быть записан ноль (тогда будет напечатано `True`), так его может и не быть (тогда ничего напечатано не будет).

Примечание: если вы планируете проверять варианты ответов у себя в терминале, обратите внимание на то, что содержащие символ `$` тексты могут изменяться при копировании – не забудьте отредактировать их в соответствии с изображением на экране. Это связано с особенностями написания `$` в некоторых видах заданий на Stepik.

Выберите все подходящие ответы из списка

Всё правильно.

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

Верно решили 23 158
учащихся

Из всех попыток 16% верных

- z ""
- !(4 -le 3)
- s \$0
- n \$0
- \$var1 == \$var2 || \$var1 != \$var2
- 5 -ge 5

[Следующий шаг](#)

[Решить снова](#)

Рис. 6.3: Варианты условий

Данный код (рис. [6.4]) проверяем на практике (рис. [6.5]). У меня вывелоось four и four (рис. [6.6]).

```
#!/usr/bin/bash
var=$1

if [[ $var -gt 5 ]]
then
    echo "one"
elif [[ $var -lt 3 ]]
then
    echo "two"
elif [[ $var -eq 4 ]]
then
    echo "three"
else
    echo "four"
fi
~
```

Рис. 6.4: Код программы в редакторе

```
[aamishina@fedora ~]$ vi prog1.sh
[aamishina@fedora ~]$ chmod +x prog1.sh
[aamishina@fedora ~]$ ./prog1.sh 3
four
[aamishina@fedora ~]$ ./prog1.sh 5
four
[aamishina@fedora ~]$ █
```

Рис. 6.5: Тестирование кода с аргументами 3 и 5

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [branching2.sh](#), [branching3.sh](#).

Посмотрите на фрагмент bash-скрипта:

```
if [[ $var -gt 5 ]]
then
    echo "one"
elif [[ $var -lt 3 ]]
then
    echo "two"
elif [[ $var -eq 4 ]]
then
    echo "three"
else
    echo "four"
fi
```

Какие строки и в какой последовательности он выведет на экран, если сначала этот скрипт запустили задав переменную `var=3`, а затем запустили еще раз, но уже с `var=5`.

Выберите один вариант из списка

Хорошая работа.

Верно решили 25 138 учащихся
Из всех попыток 64% верных

- Сначала four, потом one
- Сначала one, потом two
- Сначала two, потом one
- Сначала four, потом four

[Следующий шаг](#) [Решить снова](#)

Рис. 6.6: Ввод ответа на стеке

Для выполнения задания (рис. [6.7]) нам нужно написать код, который выводит число студентов. В данном коде программы (рис. [6.8]) мы сначала считываем число студентов в переменную `var`, а затем с помощью оператора ветвления (`if`, `elif`, `else`) выводим необходимую надпись в зависимости от числа студентов.

Напишите скрипт на bash, который принимает на вход один аргумент (целое число от 0 до бесконечности), который будет обозначать число студентов в аудитории. В зависимости от значения числа нужно вывести разные сообщения.

Соответствие входа и выхода должно быть таким:

```
0 --> No students
1 --> 1 student
2 --> 2 students
3 --> 3 students
4 --> 4 students
5 и больше --> A lot of students
```

Примечание а): выводить нужно только строку справа, т.е. "->" выводить не нужно.

Примечание б): в последней строке слово "lot" с маленькой буквы!

Примечание 2: в этой и всех последующих задачах на написание скриптов, если не указано явно, что нужно проверять вход (например, что он будет именно числом и именно от 0 до бесконечности), то этого делать **не нужно**!

Пример №1: если ваш скрипт называется `./script.sh`, то при запуске его как `./script.sh 1` на экране должно появиться:

```
1 student
```

Пример №2: если ваш скрипт называется `./script.sh`, то при запуске его как `./script.sh 5` на экране должно появиться:

```
A lot of students
```

Подсказка: в случае проблем с решением задачи, обратите внимание [на наши рекомендации по написанию скриптов](#).

Рис. 6.7: Задание

Напишите программу. Тестируется через `stdin → stdout`

Здорово, всё верно.

Верно реши
Из всех поп

Теперь вам доступен [Форум решений](#), где вы можете сравнить свое решение с другими или спросить совета.

```
1 #!/usr/bin/bash
2 var=$1
3 if [[ $var -le 0 ]]
4 then
5     echo "No students"
6 elif [[ $var -eq 1 ]]
7 then
8     echo "1 student"
9 elif [[ $var -eq 2 ]]
10 then
11    echo "2 students"
12 elif [[ $var -eq 3 ]]
13 then
14    echo "3 students"
15 elif [[ $var -eq 4 ]]
16 then
17    echo "4 students"
18 elif [[ $var -ge 5 ]]
19 then
20    echo "A lot of students"
21 fi
22
23
24
25
```

[Следующий шаг](#)

[Решить снова](#)

Рис. 6.8: Программный код

Для выполнения задания скопируем код (рис. [6.9]) в текстовый редактор и запустим файл (рис. [6.10]).

```
#!/bin/bash

for str in a , b , c_d
do
    echo "start"
    if [[ $str > "c" ]]
    then
        continue
    fi
    echo "finish"
done
```

Рис. 6.9: Программный код

```
[aamishina@fedora ~]$ vi progl.sh
[aamishina@fedora ~]$ chmod +x progl.sh
[aamishina@fedora ~]$ ./progl.sh
start
finish
start
finish
start
finish
start
finish
start
[aamishina@fedora ~]$
```

Рис. 6.10: Выполнение программы

Как мы видим, 5 раз выводится start и 4 раза finish (рис. [6.11]).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [loops1.sh](#), [loops2.sh](#).

Посмотрите на фрагмент bash-скрипта:

```
for str in a , b , c_d
do
    echo "start"
    if [[ $str > "c" ]]
    then
        continue
    fi
    echo "finish"
done
```

Если запустить этот скрипт, то сколько раз на экран будет выведено слово "start", а сколько раз слово "finish"?

Выберите один вариант из списка

Отлично!

Верно реши
Из всех поп

- 5 раз "start" и 4 раза "finish"
- 5 раз "start" и 2 раза "finish"
- 3 раза "start" и ни разу "finish"
- 3 раза "start" и 3 раза "finish"

[Следующий шаг](#)

[Решить снова](#)

Рис. 6.11: Ввод ответа на стапике

Финальное задание (рис. [6.12]) в этой мини-главе - написаное скрипта, который определяет возрастную группу пользователей (рис. [6.13]). В бесконечном цикле считываем имя, если оно пустое, то выходим. Также считываем возраст, если он 0, то выходим. Затем определяем категорию возраста group в конструкции if, elif, else и выводим сообщение с именем и возрастной группой. В конце программы выводим сообщение с прощанием bye.

Напишите скрипт на bash, который будет определять в какую возрастную группу попадают пользователи. При запуске скрипта должен вывести сообщение "enter your name:" и ждать от пользователя ввода имени (используйте `read`, чтобы прочитать его). Когда имя введено, то скрипт должен написать "enter your age:" и ждать ввода возраста (опять нужен `read`). Когда возраст введен, скрипт пишет на экран "**Имя**, your group is <группа>", где <группа> определяется на основе возраста по следующим правилам:

- младше либо равно 16: "child",
- от 17 до 25 (включительно): "youth",
- старше 25: "adult".

После этого скрипт опять выводит сообщение "enter your name:" и всё начинается по новой (бесконечный цикл!). Если в какой-то момент работы скрипта будет введено **пустое имя** или **возраст 0**, то скрипт должен написать на экран "bye" и закончить свою работу (выход из цикла).

Примеры корректной работы скрипта:

Nº1

```
./script.sh
enter your name:
Egor
enter your age:
16
Egor, your group is child
enter your name:
Elina
enter your age:
0
bye
```

Nº2:

```
./script.sh
enter your name:
Elena Petrovna
enter your age:
25
Elena Petrovna, your group is youth
enter your name:

bye
```

Рис. 6.12: Задание

```
1 # put your shell (bash) code here
2 while [[ 1==1 ]]
3 do
4     group=""
5     echo "enter your name:"
6     read name
7     if [[ -z $name ]]
8     then
9         break
10    fi
11    echo "enter your age:"
12    read age
13    if [[ $age -le 0 ]]
14    then
15        break
16    fi
17    if [[ $age -le 16 ]]
18    then
19        group="child"
20    elif [[ $age -le 25 ]]
21    then
22        group="youth"
23    else
24        group="adult"
25    fi
26    echo "$name, your group is $group"
27 done
28 echo "bye"
29
30
31
32
```

Следующий шаг

Решить снова

Рис. 6.13: Ввод ответа на стеке

7 Глава 3.4 Скрипты на bash: разное

Первое задание (рис. [7.1]). О том как складывать переменные рассказывалось в видеоуроке ранее (рис. [7.2]). Используем функцию let, без кавычек записываем через знак \$ и без пробелов, в кавычках пишем уже без знака \$, используем упрощенную форму записи $a += b$, ведь это и есть $a = a + b$.

3.4 Скрипты на bash: разное 4 из 10 шагов пройдено 1 из 14 баллов получен

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [math1.sh](#), [math2.sh](#).

Какие(ая) из предложенных ниже инструкций увеличат значение переменной `a` на значение переменной `b`? Например, если `a` было записано 10, в `b` было 5, то в `a` должно записаться 15.

Выберите все подходящие варианты!

Примечание: если вы планируете проверять варианты ответов у себя в терминале, обратите внимание на то, что содержащие символ \$ тексты могут изменяться при копировании – не забудьте отредактировать их в соответствии с изображением на экране. Это связано с особенностями написания \$ в некоторых видах заданий на Stepik.

Подсказка: обратите особое внимание на кавычки и пробелы, они могут как принципиально изменить команду, так и ни на что не повлиять (в зависимости от команды и контекста)!

Выберите все подходящие ответы из списка

Хорошая работа.

Верно решили 22 116 учащихся

Из всех попыток 20% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

- `a+=$b`
- `let "a+=b"`
- `a=$a+$b`
- `let a = a + b`
- `let a=$a+$b`

Следующий шаг

Решить снова

Рис. 7.1: Сложение переменных

Арифметика

Синтаксис:

let "переменная" = выражение"

Пример:

```
let "c" = 1 + 1
let "c" = a + b
```

Операции:

+, -, /, *	стандартные
%	остаток от деления
**	возведение в степень

Сокращение:

```
let "a=a+b" эквивалентно let "a+=b"
```

Рис. 7.2: Варианты сложения

Так как сначала мы переходим в директорию `/home/bi/`, то она и выводится по команде `pwd` (рис. [7.3]).

Вы можете скачать и изучить скрипт, который мы показали в видеофрагменте: [programs.sh](#).

Пусть вы находитесь в директории `/home/bi/Documents/` и запускаете в ней скрипт следующего содержания:

```
#!/bin/bash
cd /home/bi/
echo " pwd "
```

Что в этом случае выведет команда `echo` на экран?

Выберите один вариант из списка

Верно. Так держать!

Верно решили 23 677 учащихся
Из всех попыток 51% верных

- /home/bi
- Код возврата команды `pwd` (0 в случае успешного выполнения и не 0 в случае ошибок)
- `pwd`
- /home/bi/Documents
- pwd

Следующий шаг

Решить снова

Ваши решения Вы получили: 1 балл из 1

Рис. 7.3: Вывод пути

Далее, сначала запускаем программу, потом сверяем код возврата. Записываем в `some_file.txt` результат работы программы (рис. [7.4]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Мы рассказали, что можно проверить код возврата внешней программы прямо в конструкции `if` при помощи `if `program options arguments`` (действия внутри `if` выполняются, если программа закончилась с кодом 0). Однако это не всегда правда! Если запуск внешней программы выводит что-то в `stdout`, то в проверку if поступит именно этот вывод, а не код возврата! Вы можете убедиться в этом, написав простой bash-скрипт с использованием, например, `if `pwd``.

Однако как быть, если хочется всё-таки запустить программу `program`, которая пишет что-то в `stdout` и потом выполнить какие-то действия если ее код возврата равен 0? Выберите все верные утверждения или правильно работающие конструкции `if`.

Примечание: во всех вариантах ответов, где есть кавычка, используется именно косая кавычка (`), а не обычная () или двойная (").

Выберите все подходящие ответы из списка

Всё правильно.

Верно решили 21 426 учащихся
Из всех попыток 20% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

- Сначала запустить `program`, затем `if [[$? -eq 0]]`
- Сначала `var='program'`, затем `if [[$var -eq 0]]`
- `if `program > some_file.txt``
- `if [[`program` -eq 0]]`
- Ничего сделать нельзя

[Следующий шаг](#) [Решить снова](#)

Рис. 7.4: Задание и ответ

Для выполнения следующего задания пишем скрипт, который в цикле `for` вызывает функцию от значений от 1 до 10 (рис. [7.5]). Выполняем скрипт (рис. [7.6]) копируем ответ и вставляем его в поле ответа на стеке (рис. [7.7]).

```
#!/bin/bash
counter () {
    local let "c1+=$1"
    let "c2+=$((1)*2"
}
for i in {1..10}
do
    counter i
done
echo "counters are $c1 and $c2"
```

Рис. 7.5: Скрипт с функцией counter

```
[aamishina@fedora ~]$ vi prog2.sh
[aamishina@fedora ~]$ ./prog2.sh
counters are and 110
[aamishina@fedora ~]$ vi prog2.sh
[aamishina@fedora ~]$ █
```

Рис. 7.6: Выполнение скрипта и ответ

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [functions1.sh](#), [functions2.sh](#).

Посмотрите на функцию из bash-скрипта:

```
counter () # takes one argument
{
    local let "c1+=$1"
    let "c2+=$((1}*2"
}
```

Впишите в форму ниже строку, которую выведет на экран команда echo "counters are \$c1 and \$c2" если она находится в скрипте **после десяти вызовов** функции counter с параметрами сначала 1, затем 2, затем 3 и т.д., последний вызов с параметром 10.

Подсказка: этот пример можно решить в уме, но если система проверки не принимает ваше решение, то возможно вы что-то упустили (возможно что-то совсем небольшое/невидимое 😊). В этом случае имеет смысл написать небольшой скрипт на bash, который проделает ровно то, что указано в задании и посимвольно сверить свой ответ с тем, что он выдаст на экран.

Напишите текст

Правильно.

Верно решили 20 009 учащихся
Из всех попыток 28% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#),
отвечая на их вопросы, или сравнить свой решение с другими на [форуме решений](#).

counters are and 110

[Следующий шаг](#) [Решить снова](#)

Рис. 7.7: Задание и ответ

Следующее задание (рис. [7.8]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Напишите скрипт на bash, который будет искать наибольший общий делитель (НОД, greatest common divisor, GCD) двух чисел. При запуске ваш скрипт не должен ничего писать на экран, а просто ждет ввода двух натуральных чисел через пробел (для этого можно использовать `read` и указать ему две переменные – см. пример в видеофрагменте). После ввода чисел скрипт считает их НОД и выводит на экран сообщение "GCD is <посчитанное значение>", например, для чисел 15 и 25 это будет "GCD is 5". После этого скрипт опять входит в режим ожидания двух натуральных чисел. Если в какой-то момент работы пользователь ввел вместо этого пустую строку, то нужно написать на экран "bye" и закончить свою работу.

Вычисление НОД несложно реализовать с помощью [алгоритма Евклида](#). Вам нужно написать функцию `gcd`, которая принимает на вход два аргумента (назовем их **M** и **N**). Если аргументы равны, то мы нашли НОД – он равен **M** (или **N**), нужно выводить соответствующее сообщение на экран (см. выше). Иначе нужно сравнить аргументы между собой. Если **M** больше **N**, то запускаем ту же функцию `gcd`, но в качестве первого аргумента передаем (**M-N**), а в качестве второго **N**. Если же наоборот, **M** меньше **N**, то запускаем функцию `gcd` с первым аргументом **M**, а вторым (**N-M**).

Пример корректной работы скрипта:

```
./script.sh
10 15
GCD is 5
7 3
GCD is 1
bye
```

Примечание: в вызове функции из себя самой нет ничего страшного или неправильного, т.ч. смело вызывайте `gcd` прямо внутри `gcd`!

Примечание 2: для завершения работы функции в произвольном месте, можно использовать инструкцию `return` (все инструкции функции после `return` выполняться не будут). В отличии от `exit` эта команда завершит только функцию, а не выполнение всего скрипта целиком. Однако в данной задаче можно обойтись и без использования `return`!

Подсказка: в случае проблем с решением задачи, обратите внимание [на наши рекомендации по написанию скриптов](#).

Рис. 7.8: Задание

Ниже привожу схему, по которой работает нахождение НОДа (рис. [7.9]). Собственно этот алгоритм и описан в скрипте. Из большего числа вычитается меньшее и записывается в большее. Так происходит, пока числа не сравняются. Полученное одинаковое число и будет НОДом.

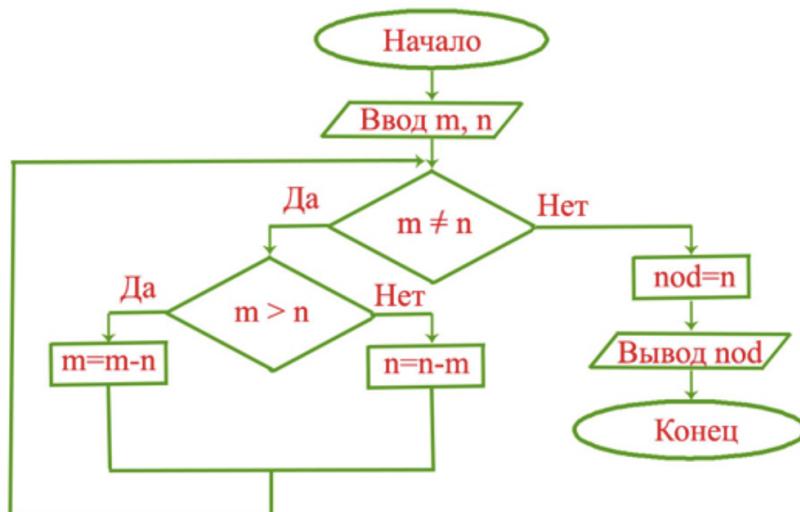


Рис. 7.9: Алгоритм из моего скрипта

Так точно!

Теперь вам доступен [Форум решений](#), где вы можете сравнить свое решение с другими или спросить совета.

```
1 # put your shell (bash) code here
2
3 gcd () # M & N
4 {
5     if [[ -z $M && -z $N ]]; then
6         echo 'bye'
7         exit
8     fi
9
10    if [[ $M -eq $N ]]; then
11        echo "GCD is $M"
12    else
13        if [[ $M -gt $N ]]; then
14            let "M-= $N"
15            gcd $M $N
16        else
17            let "N -= $M"
18            gcd $M $N
19        fi
20    fi
21 }
22 while [[ true ]]; do
23     read M N
24     gcd $M $N
25 done
```

Следующий шаг Решить снова

Рис. 7.10: Сам скрипт

Финальное задание (рис. [7.11]) - написание калькулятора с использованием конструкции switch case. Задание это несложное, так как я знаю язык программирования c++, то придумать идею реализации для меня было нетрудно. Для начала проверяем правильность введенных данных, затем выполняем действие в зависимости от знака и выводим результат (рис. [7.12]).

Напишите **калькулятор** на bash. При запуске ваш скрипт должен ожидать ввода пользователем команды (при этом на экран выводить ничего не нужно). Команды могут быть трех типов:

1. Слово "exit". В этом случае скрипт должен вывести на экран слово "bye" и завершить работу.
2. Три аргумента через пробел – первый operand (целое число), операция (одна из "+", "-", "*", "/", "%", "**) и второй operand (целое число). В этом случае нужно произвести указанную операцию над заданными числами и вывести результат на экран. После этого переходим в режим ожидания новой команды.
3. Любая другая команда из одного аргумента или из трех аргументов, но с операцией не из списка. В этом случае нужно вывести на экран слово "error" и завершить работу.

Чтобы проверить работу скрипта, вы можете записать сразу несколько команд в файл и передать его скрипту на stdin (т.е. выполнить `./script.sh < input.txt`). В этом случае он должен вывести сразу все ответы на экран.

Например, если входной файл будет следующего содержания:

```
10 + 1
2 ** 10
exit
```

то на экране будет:

```
11
1024
bye
```

Если же на вход поступит следующий файл:

```
3 - 5
2/10
exit
```

то на экране будет:

```
-2
error
```

т.к. вторая команда была **некорректной** (в ней всего один аргумент, т.к. нет пробелов между числами и операцией, а единственная допустимая команда из одного аргумента это "exit").

Рис. 7.11: Задание

Напишите программу, которая будет читать stdin и выводить результат

✓ Отличное решение!

Теперь вам доступен [Форум решений](#), где вы можете сравнить свое решение с другими или спросить совета.

Верно решили 16 980 учащихся
Из всех попыток 36% верных

```
1 #!/bin/bash
2
3 while [[ $1 == "True" ]]
4 do
5     read num1 sign num2
6     if [[ $num1 == "exit" ]]
7     then
8         echo "bye"
9         break
10    elif [[ "$num1" =~ ^[0-9]+\$ && "$num2" =~ ^[0-9]+\$ ]]
11    then
12        echo "error"
13        break
14    else
15        case $sign in
16        "+") let "result = num1 + num2";;
17        "-") let "result = num1 - num2";;
18        "/") let "result = num1 / num2";;
19        "*") let "result = num1 * num2";;
20        "%") let "result = num1 % num2";;
21        "**") let "result = num1 ** num2";;
22        *) echo "error"; break;;
23        esac
24        echo "$result"
25    fi
26 done
27
28
29
```

Следующий шаг Решить снова

Ваши решения Вы получили: 5 баллов из 5

Рис. 7.12: Мой скрипт

8 Глава 3.5 Продвинутый поиск и редактирование

Так как `iname` ищет без учета регистра, а `name` в точности как в запросе, то ответ будет следующим (рис. [8.1]):

The screenshot shows a challenge titled "3.5 Продвинутый поиск и редактирование" with a progress bar indicating "3 из 13 шагов пройдено" and "1 из 10 баллов получен". A message at the top says "Вы прошли больше 80% курса, оставьте отзыв" with buttons "Оставить отзыв" and "Нет, спасибо". The main task text reads: "Пусть в директории /home/bi лежат файлы Star_Wars.avi, star_trek_OST.mp3, STARS.txt, stardust.mpeg, Eddard_Stark_biography.txt. Отметьте все файлы, которые найдет команда `find /home/bi -iname "star*"`, но НЕ найдет команда `find /home/bi -name "star*"`?" Below this, a note says "Выберите все подходящие ответы из списка". A green checkmark next to "Хорошая работа." indicates the user's answer was correct. The list of files includes "Eddard_Stark_biography.txt", "star_trek_OST.mp3", "STARS.txt" (with a checked checkbox), "Star_Wars.avi", and "stardust.mpeg". Buttons at the bottom are "Следующий шаг" and "Решить снова". A green box at the bottom right shows statistics: "Верно решили 20 547 учащихся" and "Из всех попыток 36% верных". At the very bottom, it says "Ваши решения Вы получили: 1 балл из 1".

Рис. 8.1: Список файлов

`find` путь `-name` образец - проверяет на соответствие образцу только собственно имя файла, а `find` путь `-path` образец - проверяет на соответствие образцу полное имя файла с путем.

То есть, для файла `/home/user/test` на соответствие образцу будет проверяться: - для `-name` – только “`test`” - для `-path` – “`/home/user/test`”

Причём, для `-path` проверяется строка целиком, без учета деления на каталоги.

Соответственно выбираем правильные ответы (рис. [8.2]):

The screenshot shows a step in an advanced search and editing course. The progress bar indicates 4 из 13 шагов пройдено and 2 из 10 баллов получено. A message says 'Вы прошли больше 80% курса, оставьте отзыв' (You have completed more than 80% of the course, leave a review) with buttons 'Оставить отзыв' (Leave a review) and 'Нет, спасибо' (No, thanks). The task description asks to understand the work of options `-path` and `-name` of the `find` command and to mark all correct statements. It also mentions that users can help other participants by commenting on their answers or comparing them with others in the forum. Below the task, there is a list of four statements with checkboxes. Three are checked: 'Если заменить в команде поиска -name, на -path, то результат поиска иногда может остаться таким же', 'В некоторых случаях find с -name найдет меньше файлов, чем find с таким же запросом, но с -path', and 'В некоторых случаях find с -name найдет больше файлов, чем find с таким же запросом, но с -path'. One statement is not checked: 'Опция -path аналогична -name, но игнорирует размер букв (строчные/прописные) в имени файла'. At the bottom, there are buttons 'Следующий шаг' (Next step) and 'Решить снова' (Solve again), and a note 'Ваши решения Вы получили: 1 балл из 1' (Your solutions You got: 1 point out of 1).

Рис. 8.2: Работа опций команды `find`

Так как минимальная глубина поиска 2, то будем искать начиная от `file1` и `dir2`, а так как максимальная 3, то заканчивается все содержимым `dir2`, то есть `file2` и `dir3` (но в `dir3` мы не зайдем, это будет уже новый уровень глубины). Поэтому мы найдем только `file1` и `file2` (рис. [8.3]).

Предположим, что в директории `/home/bi/` есть следующая структура файлов и поддиректорий:

```
/home/bi/
└── dir1
    └── file1
        └── dir2
            └── file2
                └── dir3
                    └── file3
```

Какие(ой) из трех файлов (`file1`, `file2`, `file3`) будут найдены по команде `find /home/bi -mindepth 2 -maxdepth 3 -name "file*"`?

Выберите один вариант из списка

Отлично!

Верно решили 20 711 учащихся
Из всех попыток 41% верных

- Только `file3`
- Только `file2`
- Все кроме `file2`
- Ни один файл найден не будет
- Все кроме `file3`

[Следующий шаг](#)

[Решить снова](#)

[Ваши решения](#) Вы получили: 1 балл из 1

Рис. 8.3: Поиск

Из описания man “Print NUM lines of trailing context after/before matching lines”. То есть если у нас будут 1..2..10..100 строк подряд, в которых найдется совпадение, то контекст будет выведен для этой группы из 100 строк, а не до и после каждой строки в этой группе. Следовательно все размеры файлов `results.txt` будут одинаковыми (рис. [8.4]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Задание на понимание работы опций `-A`, `-B` и `-C` команды `grep`. Пусть у вас есть файл `file.txt` из 10 строк, причем в каждой строке есть слово "word". Если вы выполните на этом файле команды:

```
grep "word" file.txt > results.txt
grep -A 1 "word" file.txt > results.txt
grep -B 1 "word" file.txt > results.txt
grep -C 1 "word" file.txt > results.txt
```

то какая(ие) из них создаст файл `results.txt` наибольшего размера?

Выберите один вариант из списка

Абсолютно точно.

Верно решили 20 237 учащихся
Из всех попыток 41% верных

- grep -A 1 "word" file.txt > results.txt
- grep -C 1 "word" file.txt > results.txt
- results.txt будет одинакового размера во всех случаях
- grep -A 1 "word" file.txt > results.txt и grep -B 1 "word" file.txt > results.txt
- Все, кроме grep "word" file.txt > results.txt

Следующий шаг

Решить снова

Ваши решения Вы получили: 1 балл из 1

Рис. 8.4: Размер файлов results.txt

Объяснение на фотографии (рис. [8.5]).

ubuntu\$ - означает, что строка (а не слово) обязана заканчиваться на ubuntu.
[uU] - на этом месте должен обязательно быть один символ из списка (либо u либо U)
[xkIXKL]? - либо ничего(об этом говорит знак вопроса), либо один символ из списка(см. выше). В слове XKLubuntu будет найден один символ L.
От начала строки до [xkIXKL]? могут быть абсолютно любые символы, любые слова, любая фраза, любые предложения.
В строке "Hmm, XKLubuntu" будет найдено "Lubuntu" - и этого достаточно чтобы вывести строку на экран.

Рис. 8.5: Объяснение

Соответственно, решение задания (рис. [8.6]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Предположим, что в файле `text.txt` записаны строки, показанные среди вариантов ответа. Отметьте только те из них, которые выведет на экран команда `grep -E "[xk]XKL]?[uU]ubuntu$" text.txt`.

Выберите все подходящие ответы из списка Верно. Так держать!

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить свое решение с другими на [форуме решений](#).

Верно решили 18 768 учащихся

Из всех попыток 23% верных

- Lubuntu is better than Windows
- Lubuntu is better than Ubuntu
- Well, xubuntu is OK
- Hmm, XKUbuntu
- Linux is not always Ubuntu
- Uuuubuntu!

Ваши решения Вы получили: 2 балла из 2

Рис. 8.6: Задание и ответ на степике

Если не указывать опцию `-n`, то каждая строчка будет выведена два раза, так как “The `-n` option disables the automatic printing, which means the lines you don’t specifically tell it to print do not get printed, and lines you do explicitly tell it to print (e.g. with `p`) get printed only once” (рис. [8.7]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Что произойдет, если в команде `sed -n "/[a-z]*p" text.txt` не указывать опцию `-n`?

Выберите один вариант из списка Верно. Так держать!

Верно решили 19 784 учащихся

Из всех попыток 39% верных

- Каждая строчка будет выведена два раза
- На экран будет выведено всё содержимое файла `text.txt`
- Будут выведены все строчки файла `text.txt`, в которых есть только большие буквы латинского алфавита
- На экран ничего не напечатается

Ваши решения Вы получили: 1 балл из 1

Рис. 8.7: Вывод команды sed

Финальное задание мини-главы – написать `sed`, которая заменит все

аббревиатуры в файле на слово abbreviation. Условия вы видите на скриншоте (рис. [8.8]).

3.5 Продвинутый поиск и редактирование 12 из 13 шагов пройдено 10 из 10 баллов получено

Вы прошли больше 80% курса, оставьте отзыв Оставить отзыв Нет, спасибо

Запишите в форму ниже инструкцию sed, которая заменит все "аббревиатуры" в файле input.txt на слово "abbreviation" и запишет результат в файл edited.txt (на экран при этом ничего выводить не нужно). Обратите внимание, что в инструкции должны быть указаны и сам sed, и оба файла!

Под "аббревиатурой" будем понимать слово, которое удовлетворяет следующим условиям:

- состоит только из больших букв латинского алфавита,
- состоит из хотя бы двух букв,
- окружено одним пробелом с каждой стороны.

При этом будем считать, что в тексте **не может быть две "аббревиатуры" подряд**. Например, текст " YOU YOU and YOU!" является некорректным (в нем есть две "аббревиатуры", но они идут подряд) и на таких примерах мы проверять вашу инструкцию **не будем**.

Пример: если у вас был текст "Hi, I heard these songs by ABBA, TLA and DM!", то он должен быть преобразован в "Hi, I heard these songs by ABBA, abbreviation and abbreviation!".

Примечание: после вашей замены "аббревиатуры" на слово "abbreviation" **количество пробелов в тексте не должно меняться!**

Внимание! Во время проверки мы не запускаем команду, которую вы ввели на реальном файле с "аббревиатурами" (это небезопасно, можно же ввести rm -rf /*)! Вместо этого мы сперва анализируем структуру вашей инструкции (например, что в ней использован именно sed и сделано это ровно один раз, что на вход подается input.txt, а результат будет записан в edited.txt и т.д.), а затем запускаем её смысловую часть (т.е. поиск по регулярному выражению и замена на "abbreviation") на тестовых примерах.

К сожалению, наш запуск не идеально повторяет sed, но он очень близок к нему. Главная "несовместимость" заключается в том, что наша проверка не понимает идущие подряд символы, отвечающие за количество повторений (т.е. *, +, ? и {}). Однако эту "несовместимость" легко исправить указав при помощи "(" и ")" какой из символов к чему относится! Например, регулярное выражение a+? (ноль или один раз по одной или более букве "a") нужно записать как (a+)? (при этом запись (a)+?, конечно же, не поможет).

Рис. 8.8: Задание на степике

Мой ответ: заменяю не менее две большие латинские буквы на слово abbreviation (рис. [8.9]).

```
sed 's/[A-Z]{2,}/abbreviation /g' input.txt > edited.txt
```

Следующий шаг Решить снова

Ваши решения Вы получили: 3 балла из 3

Рис. 8.9: Ответ на степике

9 Глава 3.6 Строим графики в gnuplot

Необходимо указать опцию `-p, --persist` (можно узнать из man по gnuplot) (рис. [9.1]).

The screenshot shows a step from an online course titled "3.6 Строим графики в gnuplot". It displays a question about the `-p, --persist` option. The user has selected the correct answer, which is highlighted with a green checkmark and the message "Всё получилось!". A green box indicates that 18,785 users solved the task correctly. The question asks how to use the option to keep plots open after exiting gnuplot. The correct answer is `-p, --persist`.

Рис. 9.1: Опция gnuplot

Разобраться с этим поможет предыдущий видеоурок (рис. [9.2]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Предположим у вас есть файл `data.csv` с двумя столбцами по 10 чисел в каждом. В первой строке не записаны названия столбцов, т.е. ряды данных начинаются прямо с первой строки. Вы запускаете `gnuplot` и вводите в него две команды:

```
set key autotitle columnhead
plot 'data.csv' using 1:2
```

Какое в этом случае будет **название** у построенного ряда **данных** и сколько будет нарисовано **точек** на графике?

Выберите один вариант из списка

Отлично!

Верно решили 17 975 учащихся
Из всех попыток 32% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить свою решение с другими на [форуме решений](#).

- Название "data.csv" using 1:2", нарисовано 10 точек
- Название – первое значение из первого столбца, нарисовано 10 точек
- Название – первое значение из первого столбца, нарисовано 9 точек (точка из первой строки пропущена)
- Название – первое значение из второго столбца, нарисовано 10 точек
- Название – первое значение из второго столбца, нарисовано 9 точек (точка из первой строки пропущена)

Следующий шаг

Решить снова

Рис. 9.2: Обозначения графиков

Следующее задание заключается в написании функции `set`, которая оформляет нам номера точек и их значения (рис. [9.3]).

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Поздравляем! Вы набрали достаточно баллов для получения сертификата. Сертификат появится в вашем профиле в течение суток. [OK](#)

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [plot.gnu](#), [plot_advanced.gnu](#), [plot_advanced2.gnu](#). Все три скрипта основаны на [этой заметке](#), данные также взяты оттуда.

Предположим, что вы пишите `gnuplot`-скрипт и у вас в нем есть три переменные `x1`, `x2`, `x3`, в которых записаны координаты важных точек по оси ОХ (по возрастанию). Вы хотите, чтобы на этой си было только три деления (т.е. три черточки) в этих самых координатах, а подсчи эти делений были оформлены в виде `'point <номер точки>, value <значение соответствующей переменной>'`.

Например, для `x1=8, x2=10, x3=20`, это были бы надписи `'point 1, value 0'` в точке с координатой 0 по горизонтали, `'point 2, value 10'` в точке с координатой 10 и `'point 3, value 20'` в точке с координатой 20.

Или, например, `x1=100, x2=150, x3=250`, это были бы надписи `'point 1, value 100'` в точке с координатой 100, `'point 2, value 150'` в точке с координатой 150 и `'point 3, value 250'` в точке с координатой 250.

Впишите в форму ниже **одну команду** (т.е. одну строку), которую нужно добавить в скрипт, для выполнения этой задачи.

Примечание: проверять, что переменные `x1`, `x2`, `x3` идут по возрастанию или что они являются числами **не нужно**!

Примечание 2: в видеофрагменте на предыдущем шаге звучал термин **конкатенация**, который важен для выполнения данного задания. Под конкатенацией обычно понимают "склеивание" двух строк в одну длинную строку, например, конкатенация строк "Данные из файла " и "data.csv" даст строку "Данные из файла data.csv".

Подсказка: настоятельно рекомендуем изучить примеры скриптов – в них есть большая часть решения!

Напишите текст

Хорошая работа.

Верно решили 13 935 учащихся
Из всех попыток 44% верных

```
set xtics ("point 1, value ".x1 x1, "point 2, value ".x2 x2, "point 3, value ".x3 x3)
```

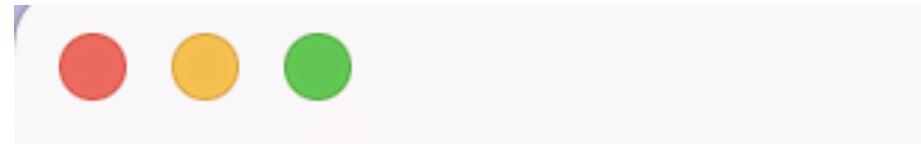
Следующий шаг

Решить снова

Рис. 9.3: Оформление точек

Для решения финального задания нам необходимо было изменить файл

согласно условию (рис. [9.4]).



```
a=a+1  
zrot=(zrot+10)%360  
set view xrot,zrot  
splot x**2+y**2  
pause 0.2  
if (a<50) reread
```

Рис. 9.4: Начальная программа

Меняем: зеркальное отражение графика ($-x^{**} 2 - y^{**} 2$), вращение изображения в обратную сторону ($(zrot+350)\%360$), скорость вращения увеличиваем в 2 раза (pause 0.1) (рис. [9.5]).

Если вы не скачали на предыдущем шаге файлы [animated.gnu](#) и [move.rot](#), то скачайте их теперь, т.к. они понадобятся для выполнения задания.

Указанные файлы использовались в последнем видеофрагменте для создания вращающегося графика. Измените инструкции в файле `move.rot` (т.е. добавлять и удалять инструкции нельзя!) таким образом, чтобы:

- График отразился зеркально относительно горизонтальной поверхности. То есть там, где была точка (10, 10, 200), станет точка (10, 10, -200), где была точка (-10, -10, 200) станет (-10, -10, -200) и т.д. При этом точка (0, 0, 0) останется на месте.
- Изображение стало вращаться в обратную сторону. То есть если раньше вращалось "влево", то теперь станет "вправо".
- Вращение стало в два раза быстрее. То есть станет в два раза больше перерисовок графика на каждую секунду вращения.

Измененный файл загрузите в форму ниже.

Примечание: наша система проверки **не может** запустить на вашем файле `move.rot` программу gnuplot и сравнить полученный график с заданным. Вместо этого мы **анализируем команды**, которые вы указали в файле. Поэтому если вы видите, что ваш скрипт в gnuplot работает точно по условию, а мы отвечаем "Incorrect/Неверно", то попробуйте упростить свою модификацию `move.rot` и отправить его еще раз.

Напишите текст

 Здорово, всё верно.

Верно решили **12 854** учащихся
Из всех попыток **47%** верных

```
a=a+1
zrot=(zrot+350)%360
set view xrot,zrot
splot -x**2-y**2
pause 0.1
if (a<50) reread
```

[Следующий шаг](#)

[Решить снова](#)

Рис. 9.5: Итоговая программа

10 Глава 3.7 Разное

В первом задании выбираем команды, которые позволяют изменить права доступа на rwxrw-r- (рис. [10.1]). Из университетского курса мы уже знаем, как это сделать (рис. [10.2]).

3.7 Разное 4 из 15 шагов пройдено 1 из 7 баллов получен

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Какая команда(ы) установят файлу file.txt права доступа rwxrw-r--, если изначально у него были права r--r--r--. Укажите **все верные** варианты ответа!

Примечание: запись вида команда1; команда2; команда3 означает, что в терминале последовательно выполнились все три команды (сначала команда1, затем команда2 и, наконец, команда3).

Выберите все подходящие ответы из списка

Правильно.

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

Верно решили 16 484 учащихся
Из всех попыток 21% верных

- chmod 764 file.txt
- chmod u+wx file.txt; chmod g+w file.txt
- chmod ug+w file.txt; chmod u+x file.txt
- chmod 467 file.txt
- chmod 777 file.txt
- chmod u-wx file.txt; chmod g-w file.txt

Следующий шаг Решить снова

Рис. 10.1: Права доступа

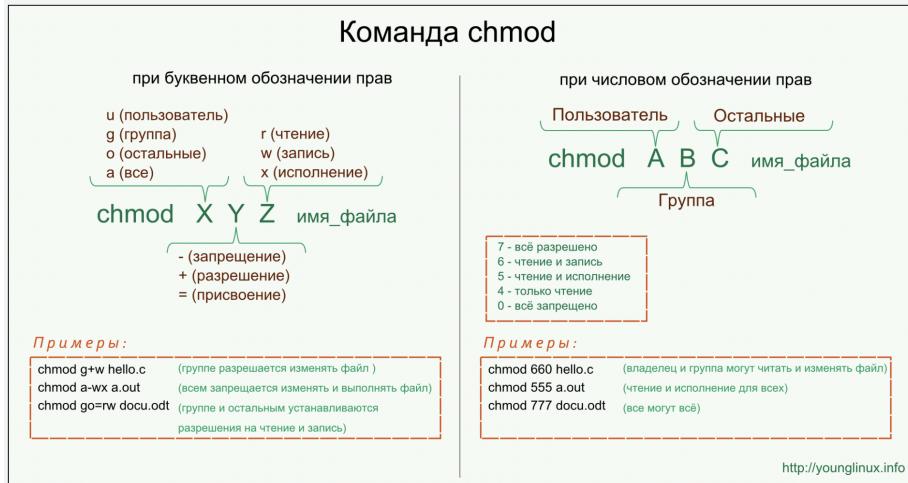


Рис. 10.2: Объяснение решения

Следующее задание (рис. [10.3]).

3.7 Разное 5 из 15 шагов пройдено 2 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв
Оставить отзыв
Нет, спасибо

Предположим вы использовали команду `sudo` для создания директории `dir`. По умолчанию для `dir` были выставлены права доступа `rwxr-xr-x` (владелец `root`, группа `root`). Таким образом никто кроме пользователя `root` не может ничего записывать в эту директорию, например, не может создавать файлы в ней.

После выполнения какой команды `user` из группы `group` всё-таки сможет создать файл внутри `dir`? Укажите **все верные** варианты ответов!

Примечание: считаем, что все команды выполняются от имени `user`, если явно не указано, что команда выполнена с `sudo`.

Примечание 2: мы выбрали пример с директорией, а не с файлом не случайно.
Дело в том, что если создать при помощи `sudo` файл с правами `rwxr--r--` в директории, которая принадлежит пользователю, то возникнет любопытная ситуация. С одной стороны пользователь может удалить этот файл (т.к. ему разрешено удалить **все** файлы внутри его директории) и может прочитать его содержимое (т.к. право "r" у файла установлено для всех), с другой стороны он не может этот файл редактировать (т.к. право "w" у файла есть только для `root`). При этом некоторые "умные" редакторы, например, `vim` позволяют даже редактировать этот файл, но сделают они это своеобразно: через удаление оригинала и создание копии уже с нужными правами (удалять мы можем, а раз можем читать, то и копию создать не сложно). Итого получается, что несмотря на права `rw-r--r--`, пользователь может сделать с этим файлом почти всё что угодно!
В случае же, когда речь идет о директории созданной `root`, ситуация будет проще: пользователь сможет смотреть её содержимое (у него есть право "r"), но удалить и создавать файлы в ней не сможет (права "w" у него нет).
Важно отметить, что **директории в Linux это в каком-то смысле файлы**. Содержимое такого "файла" – это записи о файлах и поддиректориях этой директории (такие говорят **их названия**). Таким образом, право "r" у директории дает возможность просматривать "записи", т.е. просматривать её состав. Право "w" у директории дает возможность удалять/добавлять новые "записи", т.е. удалять/создавать файлы/поддиректории в ней.
На самом деле и это еще не все. Существует так называемый **sticky bit** (атрибут файла или директории), выставление которого меняет описанное выше поведение. Файлы (или директории) с таким атрибутом смогут удалять только их владелец вне зависимости от прав, установленных у директории, в которой эти файлы (или директории) лежат!

Отдельное спасибо слушателю курса **Alexey Antipovskiy** за помощь в оформлении **Примечания 2!**

Рис. 10.3: Задание

Также из фотографии выше видно, что `a+w` - это разрешение на запись всем (рис. [10.4]).

The screenshot shows a challenge interface. At the top, there is a list of options for changing file permissions:

- sudo chmod a+w dir
- chown user:group dir
- sudo chown :group dir
- chmod o+w dir
- sudo chmod o+x dir
- sudo chmod g+w dir

Below the list are two buttons: "Следующий шаг" (Next step) and "Решить снова" (Solve again). A message below the buttons says "Ваши решения Вы получили: 1 балл из 1".

At the bottom of the interface, there is a navigation bar with "Шаг 5" and a "Следующий шаг >" button.

Рис. 10.4: Права доступа

Для следующего задания обратимся за помощью к Интернету: - wc -l вывести количество строк - wc -c вывести количество байт - wc -m вывести количество символов - wc -L вывести длину самой длинной строки - wc -w вывести количество слов

Выбираем ответ (рис. [10.5]).

The screenshot shows a challenge interface. At the top, it says "3.7 Разное 7 из 15 шагов пройдено 3 из 7 баллов получено". Below that is a message: "Вы прошли больше 80% курса, оставьте отзыв" and buttons "Оставить отзыв" and "Нет, спасибо".

There is a note: "Отметьте какие характеристики файла можно посчитать с использованием команды wc".

A section titled "Выберите все подходящие ответы из списка" contains a list of options with checkboxes:

- Правильно.
- Количество строк
- Количество символов
- Размер файла в байтах
- Количество слов
- Количество предложений

A message box says: "Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить свою решение с другими на [форуме решений](#)".

Below the list are two buttons: "Следующий шаг" and "Решить снова". A message at the bottom says "Ваши решения Вы получили: 1 балл из 1".

Рис. 10.5: Что может команда wc?

Для того, чтобы узнать, сколько места на диске занимает текущая директория, поищем информацию в man du. Опция -s - позволяет выводить размеры только для указанных явно аргументов, а не для их подкаталогов,

в `-h`, (`-human-readable`) - позволяют добавлять букву размера, например `M` для двоичного мегабайта ('мебибайт'), к каждому размеру (рис. [10.6]).

3.7 Разное 8 из 15 шагов пройдено 5 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Впишите в форму ниже команду, которая выведет сколько места на диске занимает текущая директория (при этом **размер** нужно вывести в **удобном для чтения формате** (например, вместо `2048` байт надо выводить `2.0K`) и **больше** на экран выводить **ничего не нужно**). В команде указывайте **только необходимые** для выполнения задания **опции и аргументы**, лишних опций указывать не нужно!

Пример: если в текущей директории есть два файла по `800` Кбайт и две поддиректории в каждой из которых лежит по файлу в `400` Кбайт, то загаданная команда должна вывести на экран одно число: `2.4M` (также на экране может быть выведен еще и символ `"`, обозначающий, что это размер именно текущей директории).

Напишите текст

✓ Отлично!

du -h -s

Верно решили 16 381 учащийся
Из всех попыток 53% верных

Следующий шаг Решить снова

Ваши решения Вы получили: 2 балла из 2

Рис. 10.6: Опции du

Чтобы быстро создать три директории, воспользуемся короткой формой записи `{1..3}`, которая заменит нам цикл по созданию директорий пронумерованных от 1 до 3 (рис. [10.7]).

3.7 Разное 10 из 15 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Впишите в форму максимально короткую команду (т.е. в которой минимально возможное число символов), которая позволит создать в текущей директории 3 поддиректории с именами `dir1`, `dir2`, `dir3`.

Если вы придумали команду, которая выполняет эту задачу, а система проверки сообщает вам "Incorrect"/"Неверно", то скорее всего вы придумали не самую короткую команду из возможных!

Напишите текст

✓ Хорошая работа.

mkdir dir{1..3}

Верно решили 16 720 учащихся
Из всех попыток 40% верных

Следующий шаг Решить снова

Ваши решения Вы получили: 2 балла из 2

Рис. 10.7: Быстрое создание нескольких директорий

11 Вывод

В ходе выполнения данной части курса мне рассказали о некоторых продвинутых темах в Линуксе. Я познакомилась с текстовым редактором vim, а также основами написания скриптов на bash.

К сожалению, мы не смогли выполнить все задания курса в связи с проблемами на стороне сервера стекика, но набранных мной баллов достаточно для получения сертификата, который я приложила в поле для файлов на ТУИСе (рис. [11.1]).

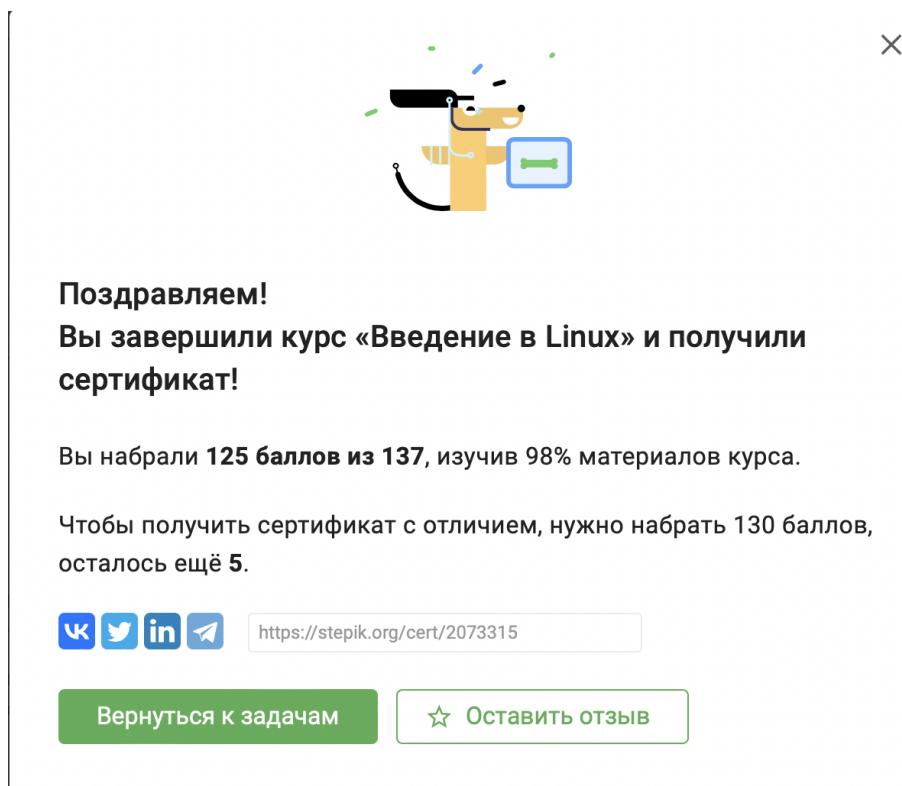


Рис. 11.1: Прохождение курса “Введение в Linux”